Expendable robots in the face of uncertainty: catastrophic failure and mortality as an information source

Emad-ud-din Muhammad[*,1], Subodh Mishra[*,2], Srikanth Saripalli[2] and Dylan Shell[1]

***Abstract*— Robots navigating hazardous environments face failures and debilitation. This paper treats a prototypical setting wherein multiple robots are tasked with performing a navigation task under uncertainty and they may fail—potentially even catastrophically. When they do fail, their failure conveys information about the world in which they operate, information of value to mission planners or overseers who may then alter their tactics subsequently. We formulate this scheme by considering a sequential deployment scenario, where each individual robot's mission is modelled as a Markov Decision Problem (MDP). Given a mission plan (i.e., a policy) along with models of mortality and communication imperfections, by analyzing the associated Markov process we describe a Bayesian estimator, constructing a likelihood (or observation model) and identifying the appropriate separation of timescales automatically in a mission independent fashion. As robots are launched sequentially from a base station, missions are executed with some successes and with some failures, allowing incremental refinement of an estimate of hazards in the world by the mission control operator. We detail a small-scale hardware implementation of the setting through the use of inexpensive UAVs, which may be rendered useless after a crash, using cheap ArUco markers for indoor positioning as a reasonable imitation of real world imperfections. The hardware demonstration is supplemented by numerous results from simulations to make the case for disposable flying vehicles.**

***Index Terms*— Expendability, Disposable Robots, Failures**

I. Introduction

Robots operating in sufficiently extreme settings, in highly contested environments, at the very edge of their performance envelopes, or under adversarial conditions will fail. Much research has aimed at maximizing robustness, seeking to avoid such outcomes and focuses on prolonging life [1]–[3]. We adopt the perspective that, even if postponed, failure is inevitable in robots and that catastrophic failure is an ineluctable fact for some scenarios. We are, therefore, interested in the question of *'How to make death count?'*

Considering a model where robot mortality is presupposed, this paper examines how destruction of a robot may be informative. The basic setting is that a mission control operator has a swarm of cheap, disposable robots that are deployed sequentially, each performing a hazardous reconnaissance mission. Robots either complete their mission successfully or fail to return. Situational awareness is improved in either case as the operator uses debilitating robot failure (or destruction) as source of information. An example of such a scenario is shown in Fig. 1.

Fig. 1: The UAV is tasked to fly from Point A to Point B and back to A again. If the UAV flies over the fan while it is on, it crashes with high probability. Success or failure to return to the launching point provides information about the operation of the fan.

Our treatment of this problem separates the concerns of the mission controller—which are high level, namely assessing the level of peril involved—from the considerations about the execution of individual missions at the low level. Both manage uncertainty: the mission control operator is solving an estimation problem, while the robots are executing a mission to actively achieve their ends (e.g., maximizing reward, reaching goal states, etc.) amid non-determinism. This paper provides a general way to construct the mission controller's estimator, given descriptions of the low level missions.

For applicability beyond one single domain, we give a pipeline of general algorithms. First we construct an individual low-level plan for an individual robot to conduct its mission, which is modelled as a Markov Decision Problem (MDP). We have UAVs flying a scouting mission in our particular case, and classical techniques are used to compute the plan (i.e., optimal policy) comprising flight actions under perturbations. This policy forms part of the input for the next stage. It is used in combination with an additional pair of models—a parameterized model describing mortality and another characterizing robot-to-base station communication—to compute a likelihood function. The resulting likelihood characterizes a virtual sensor model leading naturally to a Bayesian estimator for the mission controller. As time evolves, information from each mission is fused via this estimator to yield a sharper picture of the operating conditions. Overall, this treatment exploits a separation in timescales between the low-level policies, describing atomic actions for the robots, and their individual missions, as logical units for the mission control operator. The time leading to this separation is determined automatically in our approach.

* These authors have contributed equally.
[1] With the Department of Computer Science and Engineering.
[2] With the Department of Mechanical Engineering.
All authors are with Texas A&M University, College Station, Texas, USA.

Besides the formulation we propose to treat failure in expendable robots as an information source, we also describe a hardware demonstration using widely-available, inexpensive UAVs. Our implementation has robots executing discrete sets of actions through the use of classical techniques for continuous estimation and control. Along with hardware experiments conducted in our laboratory, we also examine several simulated scenarios.

II. Related work

Within the robotics literature, Lyu et al. [3] prefigure our work in explicitly identifying the value of expendable robots and by considering a notion of path diversity in planning motions for robots under failure. Recently, Otte and Sofge [4] treated a similar problem and adopted a perspective we discovered to be surprisingly close to ours. But contrary to their approach our model emphasizes stochasticity in the agents' missions. Also, since we focus on decoupling the operators' and robots' problems, the mission plans are provided as input for us, rather than being sought in order to maximize informativeness. Much of our paper, additionally, concerns details of the system used for our hardware demonstration. The engineering and manufacture of single-use and low-cost disposable UAVs has been considered by [5] and [6].

The policies executed by our robots realize a sort of virtual sensor. The idea that sequences of actions enable a robot to simulate a different sensor underpins the theory in [7].

III. Problem Formulation: Models & Algorithms

We organize our presentation by describing each element in detail and describe their interrelations incrementally.

A. Overview

The mission control operator, assumed to be situated at an observation outpost, has a set of identical robots that we will take to be UAVs. The robots are released sequentially and each flies their mission individually. The robots communicate accomplishment of their mission, or may communicate failure, or may fail to communicate at all. The operator, employs a Bayes filter, constructed from models of the preceding aspects, and a mortality model, to reason about the perilousness of the world. In what follows, §III-B treats an individual robot's mission; clarity of presentation is improved by treating all the UAVs as identical (so the formulation uses a single plan, but generalization to an heterogeneous setting is straightforward). Then §III-C describes our treatment of the state-based mortality model, followed by §III-D where we model a UAV's communication with the mission controller. This leads to construction of a family of MDPs (§III-E) that, under a given policy, gives a collection of Markov chains (§ III-F). Analysis of these (§ III-G) yields a mortality-informed estimator for the mission operator (§ III-H). Finally, these (many!) pieces are collected in the form of pseudocode (§ III-I).

B. The basic UAV mission

We use an MDP to describe a UAV's mission, operating under uncertain dynamics. We consider a scouting mission where a robot flies, on an outward leg, toward a point of interest, then returns to its starting point. Building on the standard definition (cf. [8]), a *mission-oriented Markov Decision Process (MDP)* is a tuple $M_{\text{mission}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, s_g \rangle$:

- The finite set of states $\mathcal{S} = \{0, \Delta x, 2\Delta x, \ldots, k\Delta x\} \times \{0, \Delta y, 2\Delta y, \ldots, \ell\Delta y\} \times \{\textit{out}, \textit{ret}\}$. We have ignored altitude and discretized the map into $k \times \ell$ spatial cells of size $\Delta x \times \Delta y$. The last part of the state encodes whether the UAV is on its outward leg or is returning.
- A set of actions $\mathcal{A} = \{\rightarrow, \leftarrow, \uparrow, \downarrow, \nwarrow, \nearrow, \swarrow, \searrow\}$ representing movement nominally toward an 8-neighbor.
- Transition function $\mathcal{P}$ written as $\mathcal{P}^a_{uv} = \mathbb{P}[\mathcal{S}_{t+1} = v | \mathcal{S}_t = u, A_t = a]$. The value *out* flips to *ret* only at the point of interest.
- The reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}^+$.
- The discount factor $\gamma \in [0, 1)$.
- A distinguished goal state $s_g \in \mathcal{S}$, with $\mathcal{P}^a_{s_g v} = 1$ if $v = s_g$, and 0 otherwise, for all $a \in \mathcal{A}$.

Notes: data collected for purposes of system identification were used to construct transitions $\mathcal{P}^a_{uv}$. Reward $\mathcal{R}$ and discount γ shape how aggressively certain map regions are to be avoided, at the cost of increased navigation. The requirement on s_g's transitions ensures it is an *absorbing* state. For our case, the UAV starts at a physical location and takes off, and then is treated as at $(0, 0, \textit{out})$. After arrival at $(0, 0, \textit{ret})$, the UAV lands, representing successful completion of the mission. All actions at $(0, 0, \textit{ret})$ are quashed, making it an absorbing state.

C. State-based mortality model

One interpretation of exponential discounting is meaningful for systems that may fail: it models circumstances in which, with probability $1 - \gamma$ at each step, something occurs to terminate the robot's execution. Unfortunately this standard treatment ties success or failure solely to the length of an execution, yielding comparatively little information. We wish, instead, to connect mortality to particular states—to do this we posit a richer model. In our model, we express factors that affect mortality via a vector $\boldsymbol{\rho} = [\rho_1, \rho_2, \ldots, \rho_\ell] \in \mathbb{R}^\ell$ of parameters.

We define a *mortality model parameterized by ℓ variables* to be a function $\mathcal{D} : \mathcal{S} \times \mathbb{R}^\ell \to [0, 1]$, where we write $\mathcal{D}(s\,;\, \rho_1, \rho_2, \ldots, \rho_\ell)$, or $\mathcal{D}(s\,;\, \boldsymbol{\rho})$, to represent the probability of catastrophic failure of a robot in state s, given that the ℓ parameters take values $\rho_1, \rho_2, \ldots, \rho_\ell$. The assumption is that the event of failure is drawn, independently, with this probability for each time step spent in state s.

Intuitively, the idea is that one identify ℓ potential threats to which to attribute reduced mortality of the robot. To each of these we attribute a severity denoted ρ_i. Then, with known values of the attributes, we can compute their contributions to the hazards facing the robot when it is in state s. (Note that there is no assumption that these

factors contribute independently.) By way of example, one might expect gusting crosswinds and radiation from a nearby reactor to affect light operations, hence positing that ρ_{wind} take a value from $0\,\text{km}$ to $50\,\text{km}$, and ρ_{rad} from $0\,\mu\text{Sv}$–$100\,\mu\text{Sv}$. Then, given a particular value of ρ_{rad} describing the source strength, the impact on the robot at $s \in \mathcal{S}$ would be scaled by its distance from the source. Similarly, given gusts with maximal magnitude ρ_{wind}, the topographic features of the environment would attenuate the effect on the robot as a function of position.

D. Modeling communication with the observation outpost

Once tasked and released, our UAVs do not maintain continuous contact with the mission control. Instead, we assume that they communicate only at a few cessation events. One example is the absorbing state s_g that the mission MDP includes as the goal. The next section will detail how, using the mortality model, an additional absorbing state, $s_\times$, is introduced for catastrophic failure. Our model of unreliable communication is simple but expressive: to each absorbing state a unique message is specified via function $\text{mesg}(\cdot)$, and also a probability of delivery via $\text{dlvr}(\cdot)$. (It is straightforward for this probability to depend on information like pose; brevity dictates we treat only the basic model here.) In our case, $s_g \overset{\text{mesg}}{\longmapsto}$ *'Mission Accomplished'*, $s_\times \overset{\text{mesg}}{\longmapsto}$ *'Mayday!'*.

E. Transformed MDP

Using M_{mission}, $\mathcal{D}(s\,;\boldsymbol{\rho})$, mesg, and dlvr, we construct a parameterized family of MDPs $M_{\text{fate}}(\boldsymbol{\rho})$ next. First, collect the absorbing states from $\mathcal{S}$ and fabricate a new state $s_\times$, which should be absorbing (i.e., extend $\mathcal{P}^a_{uv}$ so it returns to itself under all $\mathcal{A}$), giving:

$$\mathcal{S}_{\text{abs}} \coloneqq \{s \in \mathcal{S}_{\text{int}} \mid s \text{ is absorbing}\} \cup \{s_\times\}.$$

Then, given a value for $\boldsymbol{\rho}$, the MDP $M_{\text{fate}}(\boldsymbol{\rho})$ has

- States $\mathcal{S} \setminus \mathcal{S}_{\text{abs}} \cup \{s^m \mid s \in \mathcal{S}_{\text{abs}}\} \cup \{s^{\neg m} \mid s \in \mathcal{S}_{\text{abs}}\}$. In essence, each absorbing state s has been split into two states, s^m and $s^{\neg m}$, the first representing arrival of the robot at the state with $\text{mesg}(s)$ having being received by the mission controller, the second representing the case that message transmission failed.
- Construct transition function, denoted $\mathcal{T} = \mathcal{T}^a_{uv}$, as:
  (1) For all u and $v \in \mathcal{S} \setminus \mathcal{S}_{\text{abs}}$ we have, for every $a \in \mathcal{A}$, $\mathcal{T}^a_{uv} = \Big(1 - \mathcal{D}(u\,;\boldsymbol{\rho})\Big)\mathcal{P}^a_{uv}$.
  (2) For all $u \in \mathcal{S}_{\text{abs}}$ we have, also for every $a \in \mathcal{A}$, $\mathcal{T}^a_{uv} = 1$ if $u = v$ and $\mathcal{T}^a_{uv} = 0$ otherwise.
  (3) For all $u \in \mathcal{S} \setminus \mathcal{S}_{\text{abs}}$ and $v \in \mathcal{S}_{\text{abs}}$ we have:
  $$\mathcal{T}^a_{uv} = \begin{cases} \mathcal{D}(u\,;\boldsymbol{\rho})\,\text{dlvr}(v) & \text{when } v \text{ is } s^m, \\ \mathcal{D}(u\,;\boldsymbol{\rho})\,(1 - \text{dlvr}(v)) & \text{when } v \text{ is } s^{\neg m}. \end{cases}$$
- Inherit $\mathcal{A}$, $\mathcal{R}$, and γ from M_{mission}.

In other words, M_{mission} is transformed into a mortality MDP M_{fate} wherein absorbing states describe points where execution ceases, directly after a message has been sent under communication that may be unreliable. With known values for $\boldsymbol{\rho}$, the mortality parameters, we obtain a particular M_{fate}. The essence is that (1) scales the robot's dynamics so it behaves consistently with M_{mission} when it has not failed; and for failure, quantified via mortality model $\mathcal{D}(s\,;\boldsymbol{\rho})$, (3) adds treatment of message delivery. Execution may now also cease at extra states that describe the vehicle's demise $(s^m_\times, s^{\neg m}_\times)$. These two states considers a vehicle attempting to communicate in the final throes of its death, and values for $\text{dlvr}(s_\times)$ express how effective this is expected to be.

F. Decoupling policies and mission Markov chains

By construction any policy π for M_{mission} may prescribe actions in M_{fate} as well, since we only care about executions until some state in $\mathcal{S}_{\text{abs}}$ is reached. Using $M_{\text{fate}}(\boldsymbol{\rho})$ and a given π, next we can construct a family of parameterized Markov chains, which we denote $C^{[\pi]}_{\text{fate}}(\boldsymbol{\rho})$. It is the standard process: eliminate the action choice by having the transition from state s to s' depend on the actions stipulated in the policy at s, i.e., $\sum_{a\in\mathcal{A}} \Pr[\pi(s) = a]\mathcal{T}^a_{ss'}$. (This $\Pr[\cdot]$ is only needed here for polices that are not deterministic; in the deterministic case all probabilities except one are zero.) So long as either the probability of reaching s_g under π in M_{mission} is positive, or $\mathcal{D}(\cdot\,;\boldsymbol{\rho}) > 0$, then $C^{[\pi]}_{\text{fate}}(\boldsymbol{\rho})$ is an absorbing Markov chain. Typically in practice both conditions hold, though only one or the other suffices.

G. Fundamental matrix analysis of mission Markov chains

To any discrete absorbing Markov chain, with r absorbing and $n - r$ transient states, is associated a summary of expected temporal behavior that can be characterized via single $(n - r) \times (n - r)$ matrix called the *fundamental matrix* [9]. Let us denote the fundamental matrix for $C^{[\pi]}_{\text{fate}}(\boldsymbol{\rho})$ by $N(\boldsymbol{\rho})$. The entry n_{ij} of this matrix gives the expected number of times the vehicle is in transient state s_j given that it started in transient state s_i. (The most expensive step the computation of the matrix is an inversion operation.)

Two properties we require are computed, assuming the initial state of the vehicle is at s_i, as follows [9]:

- ▷ *Time to absorption* gives the expected number of steps to reach some $s \in \mathcal{S}_{\text{abs}}$: $\tau(\boldsymbol{\rho}) = \sum_{j=1}^{n-r} \text{n}_{ij}(\boldsymbol{\rho})$.
- ▷ *Absorption probabilities* give the probability that the chain is absorbed in state s_k as $\Pr(s_k, \boldsymbol{\rho}) = \sum_{j=1}^{n-r} \text{n}_{ij}(\boldsymbol{\rho})\,\text{r}_{jk}(\boldsymbol{\rho})$, where $\text{r}_{jk}(\boldsymbol{\rho})$ describes the probability of transiting from transient state s_j to absorbing state s_k, a submatrix of elements from the transition matrix of $C^{[\pi]}_{\text{fate}}$.

H. Bayesian estimation of mortality parameters

The mission control operator wishes to estimate the values of the mortality parameters $\boldsymbol{\rho}$ on the basis of missions flown by the UAVs. We treat the estimation problem via a Bayes filter. Writing z_t for the observation at time t, then a suitable update equation [10] is

$$p(\boldsymbol{\rho}|z_1, z_2, \ldots, z_t, \pi) \propto p(z_t|\boldsymbol{\rho}, \pi)\, p(\boldsymbol{\rho}|z_1, z_2, \ldots, z_{t-1}, \pi),$$

where we have been explicit in the dependence on policy π. Because, for the duration under consideration, the values of mortality parameters are assumed to be unchanging (i.e., this

is a parameter estimation problem, not a state tracking one), the recursive expression above gives the complete form of the filter once three aspects have been clarified: treatment of time, the form observations take, and the likelihood function.

1) Time (and timescale separation): To the mission operator, the significance of having not received a message depends on the time elapsed since the last UAV was dispatched. As missions involve the robot being buffeted by uncertainty, there is no well-defined point at which to conclude that it must have failed and, hence, to draw the appropriate inferences for the mortality factors. We resolve this by having the mission operator's filter be event driven: the time variable ticks when either a message is received, or we declare the vehicle to be 'missing in action.' For the latter case, we compute a time $\Delta t_{\max} = \kappa \max_{\boldsymbol{\rho}} \tau(\boldsymbol{\rho})$, where scalar κ gives a safety margin beyond the expected time to absorption. When $\mathcal{D}$ is monotonic in risk factors, as is typical, the maximum is easy to evaluate analytically, *viz.* it is just $\tau(\mathbf{0})$. Thus, analysis of the fundamental matrix helps identify an appropriate timescale to separate the mission operator abstraction from that of the low-level UAV.

2) Observations: The k^{th} observation z_k is either a message received (i.e., an element in the range of $\mathrm{mesg}(\cdot)$), or a declaration that $\Delta t_{\max}$ has elapsed and nothing has been received. We the former occurence is written as $z_k = \mathrm{mesg}(s_j)$ for some $s_j \in \mathcal{S}_{\text{abs}}$; the latter is denoted $z_k = \varnothing$.

3) Likelihood function: The sensor model, $p(z_t \,|\, \boldsymbol{\rho}, \pi)$, can now be written in terms of absorption probabilities. Concisely and exactly

$$p(z \,|\, \boldsymbol{\rho}, \pi) = \begin{cases} \Pr(s^m, \boldsymbol{\rho}) & \text{if } \exists s \in \mathcal{S}_{\text{abs}} \text{ s.t. } z = \mathrm{mesg}(s), \\ \sum_{s \in \mathcal{S}_{\text{abs}}} \Pr(s^{\neg m}, \boldsymbol{\rho}) & \text{if } z = \varnothing, \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathrm{mesg}(\cdot)$ is injective, summing across z gives 1. As the absorption probabilities are computed for the limit, the sensor model is technically only exact asymptotically for $\Delta t_{\max} \to \infty$. We express $\Delta t_{\max}$ as proportional to the expected time to absorption, because, since the actual time is distributed exponentially, a small κ can be effective.[1] The Bienaymé–Chebyshev inequality [11] easily gives a bound on the error, if desired.

I. Pseudocode of the complete algorithm

Pseudocode for the elements outlined in the preceding appears in Algorithm 1. (Note: it includes some simplifications made to match the system implementation we detail next, there $\ell = 1$, so $\boldsymbol{\rho} = \rho$ and the subscripts differential evolving estimates.)

Algorithm 1: Mortality Estimation Algorithm

Result: Estimated mortality for an agent
initialize motionModel using real flight data
initialize γ with a constant
initialize Mortality $\mathcal{D}$ and Success $\mathcal{E}$ states
initialize ρ^0 with all zero values
initialize ρ^s with an array of flat mortality profiles
$\mathcal{R} \leftarrow$ low values except for state s_g
$\mathcal{R}\{\text{indexOf}(s_g)\} \leftarrow$ high value
gridSize $\leftarrow k \times \ell$ grid size
gridRes $\leftarrow (\Delta x, \Delta y)$ resolution in meters
K $\leftarrow$ Number of planned missions
for $i \leftarrow 0$ **to** size(ρ^s) **do**
  $\mathcal{P}^s$ = evalSensorTransFunc($\rho^s\{i\}$, $\mathcal{D}$, $\mathcal{E}$, gridRes, gridSize, motionModel)
  sensorModel$\{i\}$ = computeDtmcSensorModel($\mathcal{P}^s$)
end
$\mathcal{P}^M$ = evalStateTransFunc(ρ^0, $\mathcal{D}$, $\mathcal{E}$, gridRes, gridSize, motionModel)
$\pi^M \leftarrow$ valueIterationAlgo($\mathcal{P}^M$, $\mathcal{R}$)
while $k \leq$ K **do**
  launchAgent(π^M)
  [timedOut,z_k] $\leftarrow$ retreiveMissionResult()
  if timedOut *is true* **then**
    $z_k \leftarrow \varnothing$
  end
  $Bel(\rho_{k+1})$ = bayesFilterUpdate(z_k, sensorModel, $Bel(\rho_k)$)
end

IV. The System

In this section we describe a small scale experimental setup used for demonstration. We split the computation over the UAV and a supplementary Offboard Computer owing to the UAV's physical and computational constraints; the framework is generic and can be run entirely on any UAV with adequate computational prowess.

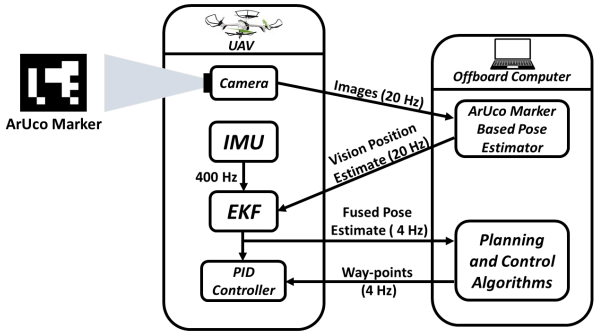


Fig. 2: System Description: Each UAV acquires images of ArUco markers and transmits data over Wi-Fi to an ArUco Marker-based Pose Estimator running on its Offboard Computer which provides a noisy measurement (*Vision Position Estimate*) of the UAV's location in space. An Extended Kalman Filter (EKF) onboard the UAV fuses the *Vision Position Estimate* with the IMU updates to generate a smooth estimate of the UAV's pose (*Fused Pose Estimate*) which is used for autonomous control and planning.

A. The UAV

The proposed approach was implemented on an inexpensive consumer UAV called the Skyviper v2450 Streaming Drone that weighs 148 g, has an average flight time of 8 min to 10 min and has a downward looking camera that transmits

[1] We used $\kappa = 2$.

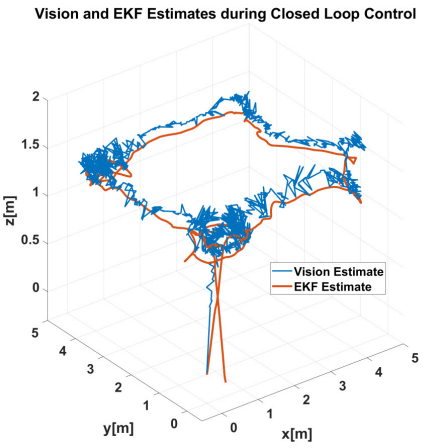


Fig. 3: A square trajectory flown using ArUco markers for localization.

images of resolution $1280\text{p}\times720\text{p}$ at $20\,\text{Hz}$[2] over Wi-Fi. It runs the latest Arducopter [12] firmware with the ability to localize itself indoors using vision-based pose[3] estimation algorithms like Visual SLAM. We used ArUco markers [13], [14]. The Skyviper's CPU runs an Extended Kalman Filter (EKF) [15] which fuses information from the onboard IMU and the ArUco Marker-based Pose Estimator (called AMPE henceforth) running on the Offboard Computer to provide a smooth estimate of the UAV's current pose. The pose estimate generated by the EKF is fed to the PID controller on board the UAV for autonomous control and way-point tracking.

B. The Offboard Computer

We use a laptop computer with ROS [16] as the Offboard Computer and communicate with the UAV over Wi-Fi using MAVROS [17]. The AMPE and the Planning algorithms run on the Offboard Computer Fig. 2 gives a visual summary of the system components and their interactions, as just described.

[2]This is an average value, in practice the frequency drops as the UAV moves farther from the Offboard Computer

[3]Position refers to (x, y, z) coordinates with respect to a frame while pose also encodes attitude in addition to position.

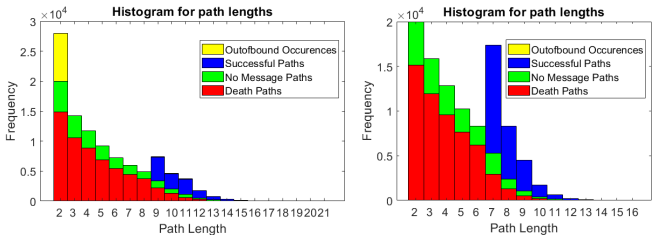


Fig. 4: Histograms of path-lengths ending in different terminal states, for 100 000 Monte-Carlo trials. Histogram for a simulated world where (Left) Start and Goal grid-cell lie on the boundary cells & (Right) Start and Goal grid-cell does not lie on the boundary cells.

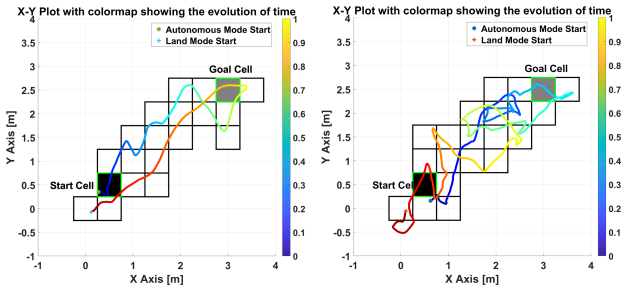


Fig. 5: UAV trajectory shown in X–Y plane. The plot shows the UAV following a two-way policy from Start Cell to Goal Cell and then back to the Start Cell (s_g) with the fan turned Off (Left frame) and On (Right frame) respectively. Increasing color shows increasing time.

V. Indoor Localization and Closed Loop Control

We carry out all the experiments indoors and localize the UAV in a planar world of ArUco markers (recall Fig. 1).

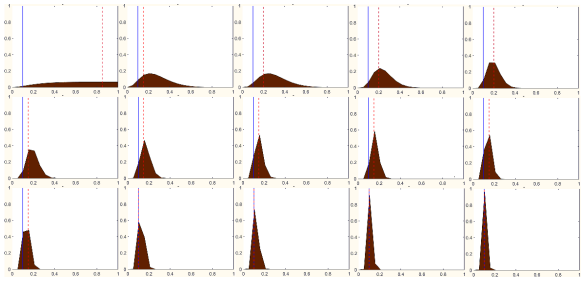
The UAV sees the markers with its downward looking camera and transmits the image over Wi-Fi to the AMPE running on the Offboard Computer at a frequency equal to the rate at which the Offboard Computer receives images from the UAV. The images are received with 300–350 ms delay, which has a direct relationship with the distance of the UAV from the Wi-FI access-point. The delay is used as tuning parameter for the EKF running onboard the UAV, which if not set close to the true value, results in oscillations in the UAV's flight, also known as 'toilet bowling'. The effect of motion blur and body vibration also deteriorate the quality of image and hence the quality of the estimated pose which ultimately results in poor closed loop control response.

For testing the AMPE, we command the UAV to follow 5 way-points, *viz.* $(0\,\text{m}, 0\,\text{m})$, $(4\,\text{m}, 0\,\text{m})$, $(4\,\text{m}, 4\,\text{m})$, $(0\,\text{m}, 4\,\text{m})$, $(0\,\text{m}, 0\,\text{m})$ which form the corners of a square in the ArUco world. The UAV negotiates a near square trajectory before landing close to where it had started (Fig. 3).

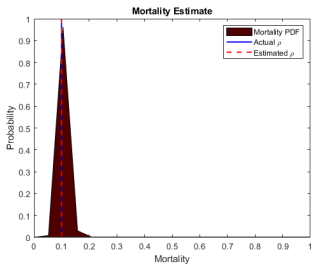
The Root Mean Squared Error (RMSE) along X, Y and Z axes were found to be $35.51\,\text{cm}$, $40.39\,\text{cm}$ and $14.29\,\text{cm}$ respectively. The RMSE was computed by comparing the EKF's pose output with the setpoints generated by the low level trajectory generator of the UAV. We use these values to tune the measurement nois parameter of the EKF onboard the UAV.

VI. Experimental Results

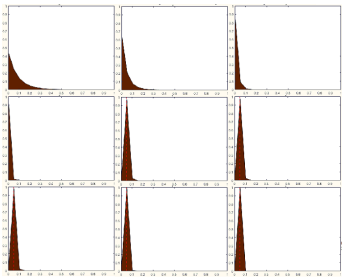
We evaluate the performance of the proposed algorithm using Monte Carlo Simulations and hardware implementation using an UAV (described in §IV and §V). An upward facing industrial fan (visible in yellow in Fig. 1) is used as a hazard in our real world experiments. The fan creates an air vortex which either leads to significant deviation of the UAV from its policy-dictated path or in a lethal crash. There are certain assumptions which are invariant between simulated and real world experiments: (1) The actions proposed by $\mathcal{A}$ remain the same for both the worlds. (2) The estimated motion model, true for the UAV, is used for both worlds.



(a) Evolution of the Bayes filter in a simulated scenario. The blue line shows the true value of ρ, known by construction in the simulator; the red dashed line is the filter's mean.



(b) An enlarged view of the final frame from Fig 6a at left. The filter has converged to the correct value $\rho = 0.1$.



(c) Analogous frames showing the filter's evolution for deployments of physical UAVs.

Fig. 6: Convergence of the mission control operator's belief about value of mortality parameter (ρ) is shown via a series of key-frames taken from the Bayes filter output in a simulation (Figs. 6a and 6b) and in physical experiments (Fig. 6c).

(3) The MDP discount-factor γ is kept at a constant 0.95 for all missions. (4) The grid size (8×8) and resolution ($0.5\,\mathrm{m}$) is kept the same for both simulated and real world experiments. (5) The same mortality model was used: it has $\ell = 1$, a single ρ parameter, that scales the spatial field constructed by placing a 2D Gaussian centered at the location of the fan.

For both worlds, we evaluate a policy π^M which takes the UAV from a starting location towards a way-point and then back towards the starting location. Policy π^M is unaware of hazards present and may cause UAV to fly straight into air vortex created by the hazard.

A. Simulation Experiments

The convergence of the estimator (in Algorithm 1) was first validated using numerous simulations because it was infeasible to reach convergence using real world experiments only, as that would require us to fly numerous missions, risking damage to the all UAVs available.[4] Apart from this, we also ran a statistically large number of simulated missions where each mission corresponded to a single Monte-Carlo trial, to evaluate the distribution of length of paths that end in terminal states and paths where agent drifted out of the boundaries of the simulation grid. These experiments can be divided into two categories: 1) trials where the start state and designated way-point were allocated cells that were close to the boundaries; 2) trials where start state and the designated way-point were allocated cells far-away from the boundaries. Results for these experiments are shown in Fig. 4. We can observe that trials for case 1 recorded a large number of cases where the agent drifted out of world boundaries. Also, trials resulting in mission success were larger for case 2 compared to case 1.

B. Real World Experiments

Given the policy π^M and the fan turned-on in the middle, *i.e.* at coordinate $(2\,\mathrm{m},2\,\mathrm{m})$, of the physical world, the experiment consists of sending the UAV from starting grid-cell to the grid-cell on the opposite side on the *out*-leg, then returning to the starting point on the *ret*-let (corresponding to s_g). This process is repeated 26 times and the corresponding flight data are logged. Out of these 26 flights, UAVs crashed 6 times and was Missing-In-Action (no message received) for a total of 3 times. We deem a UAV to have crashed when it becomes so unstable that it drops to the floor or when the standard deviation of its roll or pitch (or both) angles exceed a predefined threshold. The crash scenario occurs due to aerodynamic instability which has a detrimental effect on indoor localization (§V).

The X–Y trajectory of one of the 26 missions is shown in the right frame of Fig. 5. To make the effect of the fan on the UAV's flight path prominently perceptible, we also show a X–Y trajectory for the same policy with the fan turned off in the left frame of Fig. 5. As should be visually evident, the right frame shows a more perturbed trace than the left frame.

The evolution of the mission controller's belief (recall, we consider a case with a single ρ) is visible in Fig. 6. In simulation, in Fig. 6a, with ample data from many missions, the histogram converges to a peaked distribution around the true value (detail depicted in Fig. 6b). For hardware experiments in the real world, the set of observations produced is smaller and, though the belief shows improvement in Fig. 6c as information is gained, it would gain by additional observations. Fig. 7 shows only the maximum likelihood estimate for ρ at each time rather than the whole distribution, making it easier to see temporal aspects of the belief's evolution along with the particular messages received.

In summary, the UAV, in executing missions and failing, is able to provide information about peril within the environment. The following insights were gleaned from the experiments:

- The air vortex created by the fan, sways the UAV away from nominal path dictated by the policy and at times the UAV was not able to recover from the aerodynamic instability. This happens primarily because the UAV is unable to get a clear view of the markers and cannot localize. If this condition persists for a long time, the UAV crashes.

[4] As seen in the left frame of Fig. 6, the algorithm does converge to a value of mortality, but requires many, in this case 100, simulated missions.

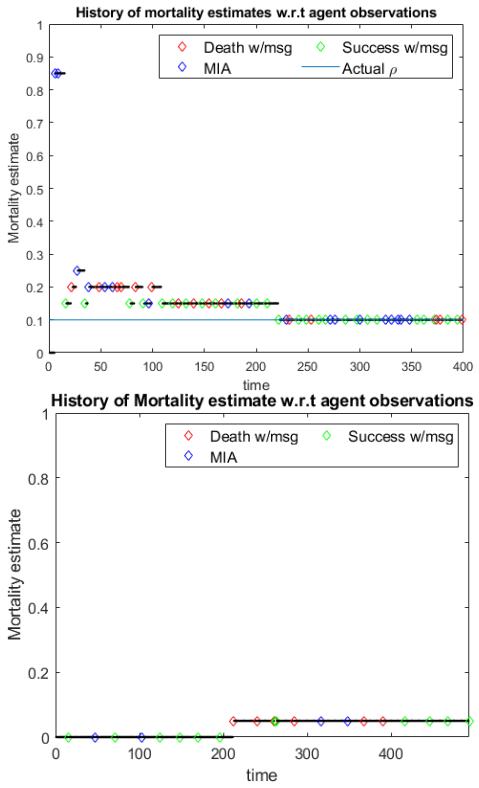


Fig. 7: Bayesian filter history of Mortality estimate over time, in (Top) a simulated world and (Bottom) physical world. Observations (z_k) are shown as colored diamonds. No diamond, signifies a point in time with no observations received. Over time, the estimate tends to converge towards a mortality estimate.

- In reality, the stochastic model of motions $\mathcal{P}$ is altered by the presence or absence of the fan, even if it does not result in complete failure. (Figure 5 shows this fact quite markedly: the robot is delayed in reaching its destination but may still succeed eventually.) The implication is that if $\Delta t_{\max}$ is computed for a motion dynamics that is benign, then the actual perturbations in the motion causes the UAV to use more time per trajectory than the time to absorption calculation factors in. This makes the mission controller believe that the UAV has gone Missing-In-Action. A value of $\kappa > 2$ helps ameliorate this effect in practice. But it points to valuable a theoretical improvement too: If one has information to estimate changes to the motion dynamics (as a function of $\boldsymbol{\rho}$), then one can easily incorporate these as modifications to $\mathcal{P}^a_{uv}$ when constructing $\mathcal{T}^a_{uv}$ for $M_{\text{fate}}(\boldsymbol{\rho})$. Doing so is straightforward and would improve the estimate of the peril present in the environment.

VII. Discussion and Future Work

We believe that one of the interesting aspects of the paper is the treatment of several elements typically confined to a particular role to be considered rather more broadly. For instance, the policies that solve an MDP are not only the output (i.e., produced by value iteration) but also serve as input to others in the longer pipeline of algorithms we describe. Automated analysis of those inputs provides an explicit time scale separation. Robots executing policies including certain communication behavior are wrapped up in *being* a sensor.

References

[1] R. C. Arkin, "Survivable Robotic Systems: Reactive and Homeostatic Control," in *Robotics and Remote Systems for Hazardous Environments*, M. Jamshidi and P. Eicker, Eds. Prentice-Hall, 1993, pp. 135–154.

[2] Y. Shapira and N. Agmon, "Path planning for optimizing survivability of multi-robot formation in adversarial environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4544–4549.

[3] Y.-H. Lyu, Y. Chen, and D. Balkcom, "k-Survivability: Diversity and Survival of Expendable Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1164–1171, Jul. 2016.

[4] M. Otte and D. Sofge, "Path Planning for Information Gathering with Lethal Hazards and No Communication," in *Proceedings of International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[5] P. E. I. Pounds, "Paper Plane: Towards Disposable Low-Cost Folded Cellulose-Substrate UAVs," in *Proceedings of the Australasian Conference on Robotics and Automation*, Dec. 2012.

[6] G. Grau, E. J. Frazier, and V. Subramanian, "Printed unmanned aerial vehicles using paper-based electroactive polymer actuators and organic ion gel transistors," *Microsystems & Nanoengineering*, vol. 2, no. 1, p. 16032, Jul. 2016.

[7] J. M. O'Kane and S. M. LaValle, "On comparing the power of robots," *International Journal of Robotics Research*, vol. 27, no. 1, pp. 5–23, Jan. 2008.

[8] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, U.S.A.: Prentice-Hall, Inc., 2009.

[9] I. Bradley and R. L. Meek, *Matrices and Society: Matrix Algebra and Its Applications in Social Sciences*. Princeton University Press, 1986.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, U.S.A.: MIT Press, 2005.

[11] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*, 2nd ed. Cambridge University Press, 2017.

[12] ArduPilot Dev Team, "Mavros," http://ardupilot.org/copter/, 2019.

[13] S. Garrido-Jurado, R. Muñoz Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, 10 2015.

[14] F. Romero Ramirez, R. Muoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, 06 2018.

[15] P. S. Maybeck, *Stochastic models, estimation, and control*, ser. Mathematics in Science and Engineering, 1979, vol. 141.

[16] M. Quigley, K. Conley, B. P Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[17] Mavlink, "Mavros," https://github.com/mavlink/mavros, 2019.